Improving the use of equational constraints in cylindrical algebraic decomposition

England, M. , Bradford, R. and Davenport, J.H.

Improving the Use of Equational Constraints in Cylindrical Algebraic Decomposition

Matthew England
Coventry University
Matthew.England@coventry.ac.uk

Russell Bradford
University of Bath
R.J.Bradford@bath.ac.uk

James H. Davenport
University of Bath
J.H.Davenport@bath.ac.uk

ABSTRACT

When building a cylindrical algebraic decomposition (CAD) savings can be made in the presence of an equational constraint (EC): an equation logically implied by a formula.

The present paper is concerned with how to use multiple ECs, propagating those in the input throughout the projection set. We improve on the approach of McCallum in ISSAC 2001 by using the reduced projection theory to make savings in the lifting phase (both to the polynomials we lift with and the cells lifted over). We demonstrate the benefits with worked examples and a complexity analysis.

Categories and Subject Descriptors

I.1.2 [**Symbolic and Algebraic Manipulation**]: Algorithms—*Algebraic algorithms, Analysis of algorithms*

General Terms

Algorithms, Experimentation, Theory

Keywords

cylindrical algebraic decomposition, equational constraint

1. INTRODUCTION

A *cylindrical algebraic decomposition* (CAD) splits $\mathbb{R}^n$ into cells arranged *cylindrically*, meaning the projections of any pair are either equal or disjoint, and such that each can be described with a finite sequence of polynomial constraints.

Introduced by Collins for quantifier elimination in real closed fields, applications of CAD include: derivation of optimal numerical schemes [18], parametric optimisation [19], epidemic modelling [9], theorem proving [27], reasoning with multi-valued functions [13], and much more.

CAD has complexity doubly exponential in the number of variables [14]. For some applications there exist algorithms with better complexity (see [2]), but CAD implementations remain the best general purpose approach for many. This may be due to the many extensions and optimisations of CAD since Collins including: partial CAD (to lift only when necessary for quantifier elimination); symbolic-numeric lifting schemes [29, 22]; local projection approaches [8, 30]; and decompositions via complex space [11, 3]. Collins original algorithm is described in [1] while a more detailed summary of recent developments can be found, for example, in [5].

ISSAC'15, July 6–9, 2015, Bath, United Kingdom.
ACM 978-1-4503-3435-8/15/07.
DOI: http://dx.doi.org/10.1145/2755996.2756678.

1.1 CAD computation and terminology

We describe the computation scheme and terminology that most CAD algorithms share. We assume a set of input polynomials (possibly derived from formulae) in ordered variables $\boldsymbol{x} = x_1 \prec \ldots \prec x_n$. The *main variable* of a polynomial (mvar) is the greatest variable present under the ordering.

The first phase of CAD, *projection*, applies projection operators repeatedly, each time producing another set of polynomials in one fewer variables. Together these contain the *projection polynomials* used in the second phase, *lifting*, to build the CAD incrementally. First $\mathbb{R}$ is decomposed into cells which are points and intervals according to the real roots of polynomials univariate in x_1. Then $\mathbb{R}^2$ is decomposed by repeating the process over each cell with the bivariate polynomials in (x_1, x_2) evaluated at a sample point.

This produces *sections* (where a polynomial vanishes) and *sectors* (the regions between) which together form the *stack* over the cell. Taking the union of these stacks gives the CAD of $\mathbb{R}^2$ and this is repeated until a CAD of $\mathbb{R}^n$ is produced.

At each stage cells are represented by (at least) a sample point and an *index*. The latter is a list of integers, with the kth describing variable x_k according to the ordered real roots of the projection polynomials in $(x_1, \ldots, x_k)$. If the integer is $2i$ the cell is over the ith root (counting from low to high) and if $2i + 1$ over the interval between the ith and $(i + 1)$th (or the unbounded intervals at either end).

The projection operator is chosen so polynomials are *delineable* in a cell: the portion of their zero set in the cell consists of disjoint sections. A set of polynomials are *delineable* if each is individually, and the sections of different polynomials are identical or disjoint. If all projection polynomials are delineable then the input polynomials must be *sign-invariant*: have constant sign in each cell of the CAD.

1.2 Equational constraints

Most applications of CAD require *truth-invariance* for logical formulae, meaning each formula has constant boolean truth value on each cell. Sign-invariance for the polynomials in a formula gives truth-invariance, but we can obtain the latter more efficiently by using equational constraints.

DEFINITION 1. *We use* QFF *to denote a quantifier free Tarski formula: Boolean combinations ($\wedge, \vee, \neg$) of statements about the signs ($= 0, > 0, < 0$) of integral polynomials.*

An equational constraint *(EC) is a polynomial equation logically implied by a QFF. If an atom of the formula it is said to be* explicit *and is otherwise* implicit.

Collins first suggested that the projection phase of CAD could be simplified in the presence of an EC [12]. He noted that a CAD sign-invariant for the defining polynomial of an EC, and sign-invariant for any others only on sections of that polynomial, would be sufficient. An intuitive approach to produce this is to consider resultants of the EC polynomial with the other polynomials, in place of them. This approach was first formalised and verified in [24].

A recent complexity analysis [5] showed that using an EC in this way reduces the double exponent in the complexity bound for CAD by 1. A natural question is whether this can be repeated in the presence of multiple ECs. An algorithm for CAD in the presence of two ECs was detailed in [25]. The main idea was to observe that the resultant of the polynomials defining two ECs is itself an EC, and so the same ideas could be applied for the second projection as for the first. However, this approach was complicated as the key result verifying [24] could not be applied recursively.

1.3 Contribution and plan

This paper discusses how we can extend the theory of ECs to produce CADs more efficiently. In Section 2.1 we revise key components of the theory for reduced projection in the presence of an EC from [24, 25]. Then in Section 2.2 we explain how it can also give reductions in the lifting phase, allowing us to propose and verify a new algorithm in Section 3 for making use of multiple ECs. This breaks with the tradition of producing CADs sign-invariant for EC polynomials, instead guaranteeing only invariance for the truth of their conjunction. We demonstrate our contributions in Sections 4 and 5 with a worked example and complexity analysis.

All experiments in MAPLE were conducted using MAPLE 18. All code and data created for this paper is openly available from `http://dx.doi.org/10.15125/BATH-00071`.

2. CAD WITH MULTIPLE EQUATIONAL CONSTRAINTS

2.1 Key theory from [23, 24, 25]

We recall some of the key theory behind McCallum's operators. Let cont, prim, disc, coeff and ldcf denote the content, primitive part, discriminant, coefficients and leading coefficient of polynomials respectively (in each case taken with respect to a given mvar). Let res denote the resultant of a pair of polynomials. When applied to a set of polynomials we interpret these as producing sets of polynomials, e.g.

$$\text{res}(A) = \{\text{res}(f_i, f_j) \,|\, f_i \in A, f_j \in A, f_j \neq f_i\}.$$

Recall that a set $A \subset \mathbb{Z}[\mathbf{x}]$ is an *irreducible basis* if the elements of A are of positive degree in the mvar, irreducible and pairwise relatively prime. Throughout this section suppose B is an irreducible basis for a set of polynomials, that every element of B has mvar x_n and that $F \subseteq B$. Define

$$P(B) := \text{coeff}(B) \cup \text{disc}(B) \cup \text{res}(B), \quad (1)$$

$$P_F(B) := P(F) \cup \{\text{res}(f, g) \mid f \in F, g \in B \setminus F\}, \quad (2)$$

$$P^*_F(B) := P_F(B) \cup \text{disc}(B \setminus F), \quad (3)$$

as the projection operators introduced respectively in [23, 24, 25]. In the general case with A a set of polynomials and $E \subseteq A$ we proceed with projection by: letting B and F be irreducible basis of the primitive parts of A and E respectively; applying the operators as defined above; and then taking the union of the output with $\text{cont}(A)$.

The theorems in this section validate the use of these operators for CAD. They use the condition of *order-invariance*, meaning each polynomial has constant order of vanishing within each cell, which of course implies sign-invariance. We say that a polynomial with mvar x_k is *nullified* over a cell in $\mathbb{R}^{k-1}$ if it vanishes identically throughout.

THEOREM 1 ([23]). *Let S be a connected submanifold of $\mathbb{R}^{n-1}$ in which each element of $P(B)$ is order-invariant.*

Then on S, each element of B is either nullified or analytic delineable (a variant on delineability, see [23]). Further, the sections of B not nullified are pairwise disjoint, and each element of such B is order-invariant on such sections.

Suppose we apply P repeatedly to generate projection polynomials. Repeated use of Theorem 1 concludes that a CAD produced by lifting with respect to these projection polynomial is order-invariant so long as no projection polynomial with mvar x_k is nullified over a cell in the CAD of $\mathbb{R}^{k-1}$ (a condition known as *well-orientedness* which can be checked during lifting). If this condition is not satisfied then P cannot be used (and we should restart the CAD construction using a different projection operator, such as Hong's [20]).

THEOREM 2 ([24]). *Let f and g be integral polynomials with mvar x_n, $r(x_1, \ldots, x_{n-1})$ be their resultant, and suppose $r \neq 0$. Let S be a connected subset of $\mathbb{R}^{n-1}$ on which f is delineable and r order-invariant.*

Then g is sign-invariant in every section of f over S.

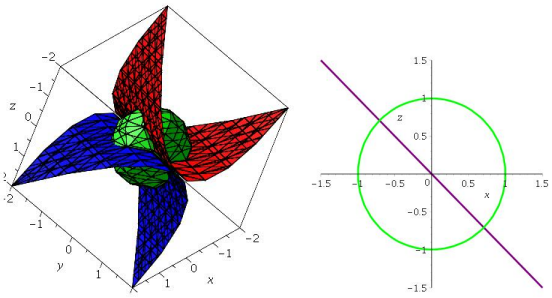
Suppose A was derived from a formula with EC defined by $E = \{f\}$, and that we apply $P_E(A)$ once and then P repeatedly to generate a set of projection polynomials. Assuming the input is well-oriented, we can use Theorem 1 to conclude the CAD of $\mathbb{R}^{n-1}$ order invariant for $P_E(A)$. The CAD of $\mathbb{R}^n$ is then sign-invariant for E using Theorem 1 and sign-invariant for A in the sections of E using Theorem 2. Hence the CAD is truth-invariant for the formula.

What if there are multiple ECs? We could designate one for special use and treat the rest as any other constraint (heuristics can help with the choice [6]). But this does not gain any more advantage than one EC gives. However, we cannot simply add multiple polynomials into E at the top level as this would result in a CAD truth-invariant for the disjunction of the ECs, not the conjunction.

Suppose we have a formula with a second EC. If this has a lower mvar then we may consider applying the reduced projection operator again at this lower level. In fact, even if the second EC is also in the mvar of the system we can *propagate* it to the lower level by noting that the resultant of the two ECs is itself an EC in one fewer variable.

So we consider applying first the operator $P_E(A)$ where E defines the first EC and then $P_{E'}(A')$ where $A' = P_E(A)$ and $E' \subseteq A'$ contains the EC in one variable fewer. Unfortunately, Theorem 2 does not validate this approach. While it could be applied once for the CAD of $\mathbb{R}^{n-1}$ it cannot then

Figure 1: The polynomials from Example 1.



validate the CAD of $\mathbb{R}^n$ because the first application of the theorem provided sign-invariance while the second requires the stronger condition of order invariance. Note however, that this approach is acceptable if $n = 3$ (since in two variables the conditions are equivalent for squarefree bases).

EXAMPLE 1. *The following are graphed in Figure 1, with g the sphere, f_1 the upper surface and f_2 the lower:*

$$f_1 = x + y^2 + z, \quad f_2 = x - y^2 + z, \quad g = x^2 + y^2 + z^2 - 1.$$

We consider the formula $\phi = f_1 = 0 \wedge f_2 = 0 \wedge g \geq 0$. The surfaces f_1 and f_2 only meet on the plane $y = 0$ and this projection is on the right of Figure 1). From this it is clear the solution requires $|x| \geq \sqrt{2}/2$ and $z = -x$.

How could this be ascertained using CAD? With variable ordering $z \succ y \succ x$ a sign-invariant CAD for (f_1, f_2, g) has 1487 cells using QEPCAD *[7]. We could then test a sample point of each cell to identify the ones where ϕ is true.*

It is preferable to use the presence of ECs. Declaring an EC to QEPCAD *will ensure it uses the algorithm in [24] based on a single use of $P_E(A)$ followed by P. Either choice results in 289 cells. In particular, the solution set is described using 8 cells: all have $y = 0, z = -x$ but the x-coordinate unnecessarily splits cells at $\frac{1}{2}(1 \pm \sqrt{6})$. This is identified due to the projection polynomial $d = \text{disc}_y(\text{res}_z(f_i, g))$.*

If we declare both ECs to QEPCAD *then it will use the algorithm in [25] applying $P_E(A)$ twice (allowed since $n = 3$) to produce a CAD with 133 cells. The solution set is now described using only 4 cells (the minimum possible). Note that d was no longer produced as a projection polynomial.*

For problems with $n > 3$ it is still possible to make use of multiple ECs. However, we must include the extra information necessary to provide order-invariance of the non-EC polynomials in the sections of ECs. The following theorem may be used to conclude that $P^*_E(A)$ is appropriate.

THEOREM 3 ([25]). *Let f and g be integral polynomials with mvar x_n, $r = \text{res}(f, g)$, $d = disc(g)$, and suppose $r, d \neq 0$. Let S be a connected subset of $\mathbb{R}^{n-1}$ on which f is analytic delineable, g is not nullified and r and d are order-invariant. Then g is order-invariant in each section of f over S.*

Suppose we have a formula with two ECs, one with mvar x_n and the other with mvar x_{n-1}. The second could be explicit in the formula or implicit (a resultant as described earlier). Theorem 3 allows us to use a reduced operator twice. We first calculate $A' = P_E(A)$ where E contains the defining polynomial of the first EC, and then $P^*_{E'}(A')$ where E' contains the defining polynomial of the other. Subsequent projections simply use P. When lifting we use Theorem 1 to verify the CAD of $\mathbb{R}^{n-2}$ as order-invariant for $P^*_{E'}(A')$; Theorem 1 to verify the CAD of $\mathbb{R}^{n-1}$ order-invariant for E' everywhere and Theorem 3 to verify it order-invariant for A' in the sections of E'; and Theorem 1 and 2 to verify the CAD of $\mathbb{R}^n$ order-invariant for E and sign-invariant for A in those cells that are both sections of E and E'.

2.2 Reductions in the lifting phase

The main contribution of the present paper is to realise that the theorems above also allow for significant savings in the lifting phase of CAD. However, to implement these we must discard two embedded principles of CAD:

1. That the projection polynomials are a fixed set.
2. That the invariance structure of the final CAD can be expressed in terms of sign-invariance of polynomials.

Abandoning the first is key to recent work in [11, 3], while the second was also investigated in [10, 26].

2.2.1 Minimising the polynomials when lifting

Consider Theorem 2: it allows us to conclude that g is sign-invariant in the sections of f produced over a CAD of $\mathbb{R}^{n-1}$ order-invariant for $P_{\{f\}}(\{f, g\})$. Therefore, it is sufficient to perform the final lift with respect to f only (decompose cylinders according to the roots of f but not g). The decomposition imposes sign-invariance for f while Theorem 2 guarantees it for g in the cells where it matters.

EXAMPLE 2. *We return to Example 1. Recall that designating either EC and using [24] produced a CAD with 289 cells. If we follow this approach but lift only with respect to the designated EC at the final step (implemented in our* MAPLE *package [17]) we obtain a CAD with 141 cells.*

This improved lifting follows from the theorems in [24], but was only noticed 15 years later during the generalisation of [24] to the case of multiple formulae in [4, 5]. Experiments there demonstrated its importance, particularly for problems with many constraints (see Section 8.3 of [5]).

When we apply a reduced operator at two levels then we can make such reductions at both the corresponding lifts.

EXAMPLE 3. *We return to the problem from Example 1. Set $A = \{f_1, f_2, g\}$ and $E = \{f_1\}$. Then project out z using*

$$P_E(A) = \{y^2, y^4 + 2xy^2 + 2x^2 + y^2 - 1\}.$$

These are the resultants of f_1 with f_2 and g. The discriminant of f_1 was a constant and so could be discarded, as was its leading coefficient (meaning no further coefficients were required). We set $A' = P_E(A)$, $E' = \text{res}_z(f_1, f_2) = y^2$ and

$$R = \text{res}_y(y^2, y^4 + 2xy^2 + 2x^2 + y^2 - 1) = (2x^2 - 1)^2.$$

We have $P_{E'}(A') = \{R\}$ since the other possible entries (the discriminants and coefficients from E') are all constants. We hence build a 5 cell CAD of the real line with respect to the two real roots of R. We then lift above each cell with respect to y^2 only, in each case splitting the cylinder into three cells about $y = 0$, to give a CAD of $\mathbb{R}^2$ with 15 cells.

Finally, we lift over each of these 15 cells with respect to f_1 to give 45 cells of $\mathbb{R}^3$. This compares to 133 from QEPCAD*, which used reduced projection but then lifted with all projection polynomials. No polynomials were nullified, so using Theorems 1 and 2, the output is truth-invariant for ϕ.*

The additional lifting that QEPCAD performed does not provide any further structure. For example, if we had lifted with respect to f_2 at the final stage in Example 3 then we would be doing so without the knowledge that it is delineable. Hence splitting the cylinder at the sample point offers no guarantee that the cells produced are sign-invariant away from that point. So the extra work does not allow us to conclude that f_2 is sign-invariant (except on sections of f_1).

Note that using fewer projection polynomials for lifting not only decreases output size (and computation time) but also the risk of failure from non well-oriented input: we only need worry about nullification of polynomials we lift with.

2.2.2 Minimising the cells for stack generation

We can achieve still more savings from the theory in Section 2.1 by abandoning the aim of producing a CAD sign-invariant with respect to any polynomial, instead insisting only on truth-invariance for the formula. We may then lift trivially to cylinders over cells already known to be false, only identifying sections of projection polynomials if there is a possibility the formula may be true. The idea of avoiding computations over false cells was presented in [28]. Our contribution is to explain how such cells can easily be identified in the presence of ECs. We demonstrate with our example.

EXAMPLE 4. *Return to the problem from Examples 1 − 3 and in particular the CAD of $\mathbb{R}^2$ produced with 15 cells in Example 3. On 5 of these 15 cells the polynomial R is zero and on the others it is either positive or negative throughout.*

Now, ϕ can only be satisfied above the 5 cells, as elsewhere the two EC defining polynomials cannot share a root and thus vanish together. We can already conclude the truth value for the 10 cells (false) and thus we do not need to lift over them, except in the trivial sense of extending them to a cylinder in $\mathbb{R}^3$. Lifting over the 5 cells where $R = 0$ with respect to f_1 gives 15 cells, which combined with the 10 cylinders gives a CAD of $\mathbb{R}^3$ with 25 cells that is truth-invariant for ϕ.

This 25 cell CAD is not sign-invariant for f_1. The cylinders above the 10 cells in $\mathbb{R}^2$ where $R \neq 0$ may have f_1 varying sign, but since f_1 can never equal zero at the same time as f_2 in these cells it does not affect the truth of ϕ.

Identifying the 5 cells where $R = 0$ in the CAD of $\mathbb{R}^2$ was trivial since they are simply the sections of the second lift, and hence those cells with second entry even in the cell index. Those sections produced in the third lift are similarly all cells where f_1 is zero, however, we cannot conclude that f_2 is also zero on these. Theorem 2 only guarantees that f_2 is sign-invariant on such cells, so to determine those signs we must still evaluate the polynomials at the sample point.

Reducing the number of cells for stack generation clearly decreases output size, and since the cells can be identified using only a parity check on an integer, computation time decreases also. As with the improvements in Section 2.2.1, this also decreases the risk of non well-oriented input: we only need worry about nullification over these identified cells.

3. ALGORITHM

We present Algorithm 1 to build a truth-invariant CAD for a formula in the presence of multiple ECs. We assume that the ECs are already identified as input to the algorithm (they may have been first computed through propagation as described in Section 2). We assume further that each EC is primitive, and that all the ECs have different mvar (so in practice a choice of designation may have been made).

Algorithm 1: CAD using multiple ECs

Input : A formula ϕ in variables $x_1, \ldots, x_n$, and a sequence of sets $\{E_k\}_{k=1}^n$; each either empty or containing a single primitive polynomial with mvar x_k which defines an EC for ϕ.

Output: Either: $\mathcal{D}$, a truth-invariant CAD of $\mathbb{R}^n$ for ϕ (described by lists I and S of cell indices and sample points); or **FAIL**, if not well-oriented.

1 Extract from ϕ the set of defining polynomials A_n;
2 **for** $k = n, \ldots, 2$ **do**
3 Set B_k to the finest squarefree basis for prim(A_k);
4 Set C to cont(A_k);
5 Set F_k to the finest squarefree basis for E_k;
6 **if** F_k *is empty* **then**
7 Set $A_{k-1} := C \cup P(B_k)$;
8 **else**
9 **if** $k = n$ *or* $k = 2$ **then**
10 Set $A_{k-1} := C \cup P_{F_i}(B_i)$;
11 **else**
12 Set $A_{k-1} := C \cup P^*_{F_i}(B_i)$;
13 If E_1 is not empty then set p to be its element; otherwise set p to the product of polynomials in A_1;
14 Build $\mathcal{D}_1 := (I_1, S_1)$ according to the real roots of p;
15 **if** $n = 1$ **then**
16 **return** $\mathcal{D}_1$;
17 **for** $k = 2, \ldots, n$ **do**
18 Initialise $\mathcal{D}_k = (I_k, S_k)$ with I_k and S_k empty sets;
19 **if** F_k *is empty* **then**
20 Set $L := B_k$;
21 **else**
22 Set $L := F_k$;
23 **if** E_{k-1} *is empty* **then**
24 Set $\mathcal{C}_a := \mathcal{D}_{k-1}$ and $\mathcal{C}_b$ empty;
25 **else**
26 Set $\mathcal{C}_a$ to be cells in $\mathcal{D}_{k-1}$ with $I_{k-1}[-1]$ even;
27 Set $\mathcal{C}_b := \mathcal{D}_{k-1} \setminus \mathcal{C}_a$;
28 **for** *each cell* $c \in \mathcal{C}_a$ **do**
29 **if** *An element of* L *is nullified over* c **then**
30 **return** FAIL;
31 Generate a stack over c with respect to the polynomials in L, adding cell indices and sample points to I_k and S_k;
32 **for** *each cell* $c \in \mathcal{C}_b$ **do**
33 Extend to a single cell in $\mathbb{R}^k$ (cylinder over c), adding index and sample point to I_k and S_k;
34 **return** $\mathcal{D}_n = (I_n, S_n)$.

Steps 1 − 12 run the projection phase of the algorithm. Each projection starts by identifying contents and primitive parts. When there is no declared EC (E_i is empty) the projection operator (1) is used (step 7). Otherwise the operator (3) is used (step 12), unless it is the very first or very last projection (step 10) when we use (2). In each case the output of the projection operator is combined with the contents to form the next layer of projection polynomials.

Steps 13 − 16 construct a CAD for the real line (and return it if the input was univariate). This is sometimes referred to in the literature as the *base phase*. If there is a declared EC in the smallest variable then the real line is decomposed according to its roots, otherwise according to the roots of all the univariate projection polynomials.

Steps 17 − 33 run the lifting phase, incrementally building CADs of $\mathbb{R}^k$ for $k = 2, \ldots, n$. For each k there are two considerations. First, whether there is a declared EC with mvar x_k. If so we lift only with respect to this (step 22) and if not we use all projection polynomials with mvar x_k (step 20). Second, whether there is a declared EC with mvar x_{k-1}. If so we restrict stack generation to those cells where the EC was satisfied. These are simply those with $I_{k-1}[-1]$ (last entry in the cell index) even (step 26). We lift the other cells trivially to a cylinder in step 33.

Algorithm 1 clearly terminates. We will verify that it produces a truth-invariant CAD for the formula so long as the input is well-oriented, as defined below.

DEFINITION 2. *For $k = 2, \ldots, n$ define sets:*

- *L_k − the* lifting polynomials*: the defining polynomial of the declared EC with mvar x_k if one exists, or all projection polynomials with mvar x_k otherwise.*
- *$\mathcal{C}_k$ − the* lifting cells*: those cells in the CAD of $\mathbb{R}^{k-1}$ in which the designated EC with mvar x_{k-1} vanishes if it exists, and all cells in that CAD otherwise.*

The input of Algorithm 1 is well-oriented *if for $k = 2, \ldots, n$ no element of L_k is nullified over an element of $\mathcal{C}_k$.*

THEOREM 4. *Algorithm 1 satisfies its specification.*

PROOF. We must show the CAD is truth-invariant for ϕ, unless the input is not well-oriented when FAIL is returned.

First consider the case where $n = 1$. The projection phase would not run, with the algorithm jumping to the CAD construction in step 13, returning the output in step 16. If there was no declared EC then the CAD is sign-invariant for all polynomials defining ϕ and thus every cell is truth invariant for ϕ. If there was a declared EC then the output is sign-invariant for its defining polynomial. Cells would either be intervals where the formula must be false; or points, where the EC is satisfied, and the formula either identically true or false depending on the signs of the other polynomials.

Next suppose that the input were not well-oriented (Definition 2). For a fixed k, the conditional in steps 19 − 22 sets the lifting polynomials L_k to L and the conditional in steps 23 − 27 the lifting cells $\mathcal{C}_k$ to $\mathcal{C}_a$. Thus it is exactly the conditions of Definition 2 which are checked by step 29, returning FAIL in step 30 when they are not satisfied. If the lifting phase completes then the input is well-oriented.

From now on we suppose $n > 1$ and the input is well-oriented. For a fixed k define *admissible* cells to be those in the induced CAD of $\mathbb{R}^{k-1}$ where all declared ECs with mvar smaller than x_k are satisfied, or to be all cells in that induced CAD if there are no such ECs. Then let $I(k)$ be the following statement for the CADs produced by Algorithm 1. *Over admissible cells (in $\mathbb{R}^{k-1}$) the CAD of $\mathbb{R}^k$ is:*

(a) order-invariant for any EC with mvar x_k;
(b) order- (sign- if $k = n$) invariant for all projection polynomials with mvar x_k on sections of the EC over admissible cells, or over all admissible cells if no EC exists.

We have already proved $I(1)$, and $I(n)$ may be proved by induction. To assert the truth of $I(k)$ we note the following:

- When E_k is empty we use Theorem 1 to assert all projection polynomials with mvar x_k are order-invariant in the stacks over admissible cells giving (a) and (b).
- When E_k is not empty and $k = 2$ we used the projection operator (2). Theorem 2 allows us to conclude (b) and that the EC is sign-invariant in admissible cells. The stronger property of order-invariance follows automatically since the lifting polynomials form a squarefree basis in two variables.
- When E_k is not empty and $k = n$ we used the projection operator (2). Theorem 2 allows us to conclude (b), but also (a) since in the case $k = n$ the statement requires only sign-invariance.
- When E_k is not empty and $2 < k < n$ we used the projection operator (3). Theorem 3 then allows us to conclude the statement.

In each case the assumptions of the theorems are met by the inductive hypothesis, exactly over admissible cells as defined according to whether E_{k-1} was empty or not.

From the definition of admissible cells, we know that ϕ is false (and thus trivially truth invariant) upon all cells in the CAD of $\mathbb{R}^n$ built over an inadmissible cell of $\mathbb{R}^k$, $k < n$. Coupled with the truth of (a) for $k = 1, \ldots, n$, this implies the CAD of $\mathbb{R}^n$ is truth-invariant for the conjunction of ECs (although it may not be truth-invariant for any one individually). The truth of (b) implies that on those cells where all ECs are satisfied, the other polynomials in ϕ are sign-invariant and thus ϕ is truth-invariant. □

4. WORKED EXAMPLE

Assume variable ordering $z \succ y \succ x \succ u \succ v$ and define

$$f_1 := x - y + z^2, \quad f_2 := z^2 - u^2 + v^2 - 1, \quad g := x^2 - 1,$$
$$f_3 := x + y + z^2, \quad f_4 := z^2 + u^2 - v^2 - 1, \quad h := z.$$

We consider the formula

$$\phi = f_1 = 0 \wedge f_2 = 0 \wedge f_3 = 0 \wedge f_4 = 0 \wedge g \geq 0 \wedge h \geq 0.$$

The solution can be found manually by decomposing the system into blocks. The surfaces f_1 and f_3 are graphed in (x, y, z)-space on the left of Figure 2. They meet only on the plane $y = 0$ and this projection is shown on the right. The surfaces f_2 and f_4 are graphed in (z, u, v)-space on the left of Figure 3 and meet only when $z = \pm 1$. We consider only $z = +1$ due to $h \geq 0$, with this projection plotted on the right. We thus see that the solution set is given by

$$\{u = \pm v, x = -1, y = 0, z = 1\}.$$

To ascertain this by Algorithm 1 we must first propagate and designate ECs. We choose to use f_1 first, calculate

$$\mathrm{res}_z(f_1, f_2) = (-u^2 + v^2 - x + y - 1)^2$$

and assign r_1 to be the square root: the defining polynomial for an EC with mvar y. Similarly consider

$$\mathrm{res}_y\left(r_1, \mathrm{res}_z(f_1, f_3)\right) = 16(u^2 - v^2 + x + 1)^4,$$
$$\mathrm{res}_y\left(r_1, \mathrm{res}_z(f_1, f_4)\right) = 4(u^2 - v^2)^2$$

and assign $r_2 := u^2 - v^2 + x + 1$, $r_3 := u^2 - v^2$: defining polynomials for ECs with mvar x and u respectively. There is no series of resultants that leads to an EC with mvar u. We hence have $\{E_j\}_{k=1}^n := \{f_1\}, \{r_1\}, \{r_2\}, \{r_3\}, \{\,\}$ as input for Algorithm 1, along with ϕ.

Figure 2: The polynomials f_1 and f_3 from Section 4.

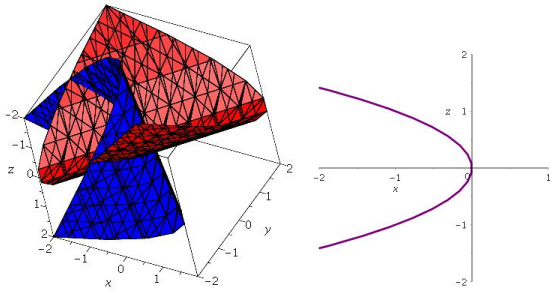
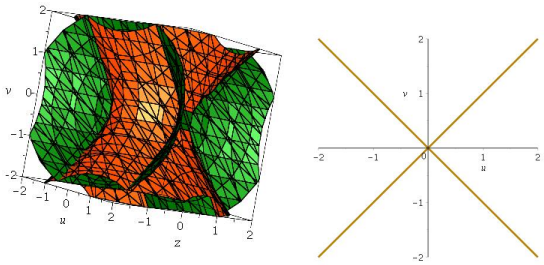


Figure 3: The polynomials f_2 and f_4 from Section 4.



The algorithm starts by extracting the defining polynomials $A_5 = \{f_1, f_2, f_3, f_4, g, h\}$ and finds $B_5 = A_5, F_5 = E_5$ (in fact $F_i = E_i$ for all $i = 1, \ldots, 5$). There is a declared EC for the first projection so we use the operator (2) to derive

$$A_4 := P_{F_5}(B_5) = \{(x^2-1)^2, (-u^2+v^2-x+y-1)^2, (u^2-v^2-x+y-1)^2, 4y^2, x-y\}.$$

Hence $C := \{x^2-1\}$ and

$$B_4 := \{y, y-x, -u^2+v^2-x+y-1, u^2-v^2-x+y-1\}.$$

For the next projection we must use operator (3), giving

$$A_3 := C \cup P^*_{F_4}(B_4) = \{x^2-1, u^2-v^2+x+1, u^2-v^2, u^2-v^2+1\}$$

noting that for this example the extra discriminants in (3) all evaluated to constants and so could be discarded. Then

$$B_3 := \{x^2-1, u^2-v^2+x+1\}, \quad C := \{u^2-v^2, u^2-v^2+1\},$$

and the next projection also uses (3) to produce

$$A_2 := \{u^2-v^2, u^2-v^2+1, u^4-2u^2v^2+v^4+2u^2-2v^2\}.$$

For the final projection there is no EC and so we use operator (1) to find $A_1 := \{v^2\}$. The base phase of the algorithm hence produces a 3-cell CAD of the real line isolating 0.

For the first lift we have $L = \{u^2-v^2\}$ and C_a containing all 3 cells. Above the two intervals we split into 5 cells by the curves $u = \pm v$, while above $v = 0$ we split into three cells about the origin. From these 13 cells of $\mathbb{R}^2$ we select the 5 which were sections of $u^2 - v^2$ for C_a. These are lifted with respect to $L = \{r_2\}$, and the other 8 are simply extended to cylinders in $\mathbb{R}^3$. Together this gives a CAD of $\mathbb{R}^3$ with 23 cells. The next two lifts are similar, producing first a CAD of $\mathbb{R}^4$ with 53 cells and finally a CAD of $\mathbb{R}^5$ with 113 cells. The entire calculation takes less than a second in Maple.

Choice in EC designation

Algorithm 1 could have been initialised with alternative EC designations. There were the 4 explicit ECs with mvar z, and by taking repeated resultants we discover the following implicit ECs, organised in sets with decreasing mvar:

$$\{y^2, u^2-v^2+x-y+1, -u^2+v^2+x-y+1, u^2-v^2+x+y+1, -u^2+v^2+x+y+1\},$$
$$\{x+1, -u^2+v^2+x+1, u^2-v^2+x+1\}, \quad \{u^2-v^2\}.$$

There are hence 60 possible permutations of EC designation, but they lead to only 3 different outputs, with 113, 103 and 93 cells. Heuristics for other questions of CAD problem formulation [15, 6, 21, 31] could likely be adapted to assist here. We note that 93 cells is not a minimal truth invariant CAD for ϕ as it splits the CAD of $\mathbb{R}^1$ at $v = 0$ (identified from the discriminant of the only EC with mvar u).

Comparison with other CAD implementations

A sign-invariant CAD of $\mathbb{R}^5$ for the 6 polynomials in the example could be produced by Qepcad with 1,118,205 cells. Neither the `RegularChains` Library in Maple [11] nor our Maple package [17] could produce one in under an hour.

Our implementation of [24], which uses operator (2) once but also performs the final lift with respect to the EC only, can produce a CAD with either 3023, 10935 or 48299 (twice) cells depending on which EC is designated. The Qepcad implementation of [24] gives 11961, 30233, 158475, or 158451 cells. Comparing these sets of figures we see the dramatic improvements from just a single reduced lift.

Allowing Qepcad to propagate the 4 ECs (so a similar projection phase as Algorithm 1 but then a normal CAD lifting phase) produces a CAD with 21079 cells. By declaring only a subset of the 4 (which presumably changes the designations of implicit ECs) a CAD with 5633 cells can be produced, still much more than using Algorithm 1.

The `RegularChains` Library can also make use of multiple ECs, as detailed in [3]. The version in Maple 18 times out after an hour, however, with the development version a CAD can be produced instantly. There are choices (with analogies to designation [16]) but they all lead to a 137 cell output. In particular, they all have an induced CAD of the real line which splits at $v = \pm 1$ as well as $v = 0$.

5. COMPLEXITY ANALYSIS

We build on recent work in [5] to measure the dominant term in bounds on the number of CAD cells produced. Numerous studies have shown this to be closely correlated to the computation time. We assume input with m polynomials of maximum degree d in any one of n variables.

Definition 3. *Consider a set of polynomials p_j. The* combined degree *of the set is the maximum degree (taken with respect to each variable) of the product of all the polynomials in the set:* $\max_i(\deg_{x_i}(\prod_j p_j))$.

The set has the (m,d)-property *if it may be partitioned into m subsets, each with maximum combined degree d.*

For example, $\{y^2 - x, y^2 + 1\}$ has combined degree 4 and thus the $(1, 4)$-property, but also the $(2, 2)$-property.

This property (introduced in McCallum's thesis) can measure growth in the projection phase. In [5] we proved that if A has the (m, d)-property then $P(A) \cup \text{cont}(A)$ has the $(M, 2d^2)$-property with $M = \lfloor \frac{1}{2}(m+1)^2 \rfloor$. When $m > 1$, we can bound M by m^2 (but we need $2m^2$ to cover $m = 1$).

If A has the (m,d)-property then so does its squarefree basis. Hence applying this result recursively (as in Table 1) measures the growth in (m,d)-property during projection under operator (1). After the first projection there are multiple polynomials and so the tighter bound for M is used.

The number of real roots in a set with the (m,d)-property is at most md. The number of cells in the CAD of $\mathbb{R}^1$ is thus bounded by twice the product of the final two entries, plus 1. Similarly, the total number of cells in the CAD of $\mathbb{R}^n$ by

$$(2md+1)\prod_{r=1}^{n-1}\left[2\left(2^{2^{r-1}}m^{2^r}\right)\left(2^{2^r-1}d^{2^r}\right)+1\right]. \quad (4)$$

Omitting the +1s will leave us with the dominant term of the bound, which evaluates to give the following result.

THEOREM 5. *The dominant term in the bound on the number of CAD cells in $\mathbb{R}^n$ produced using (1) is*

$$(2d)^{2^n-1}m^{2^n-1}2^{2^{n-1}-1}. \quad (5)$$

From now on assume ℓ ECs, $0<\ell\leq\min(m,n)$, all with different mvar. For simplicity we assume these variables are $x_n,\ldots,x_{n-\ell+1}$ (the first ℓ projections are reduced).

LEMMA 6. *Suppose A is a set with the (m,d)-property and $E\subset A$ has the $(1,d)$-property. Then $\mathrm{cont}(A)\cup P_E^*(A)$ has the $(2m,2d^2)$-property.*

PROOF. In [5] we proved that applying $P_E(A)\cup\mathrm{cont}(A)$ gives a set of $\left\lfloor\frac{1}{2}(3m+1)\right\rfloor$ polynomials of combined degree $2d^2$. The extra $m-1$ discriminants required by operator (3) will each have degree at most $d(d-1)$, so pairing them we have $\lceil\frac{1}{2}(m-1)\rceil$ sets of combined degree at most $2d^2$. Then

$$\left\lfloor\tfrac{1}{2}(3m+1)\right\rfloor+\lceil\tfrac{1}{2}(m-1)\rceil=m+\left\lfloor\tfrac{1}{2}(m+1)\right\rfloor+\left\lfloor\tfrac{m}{2}\right\rfloor$$

and since $m\in\mathbb{Z}$ this always equals $2m$. □

We apply this recursively in the top half of Table 2, with the bottom derived via the process for P, as in Table 1.

Define d_i and m_i as the entries in the Number and Degree columns of Table 2 from the row with i Variables. We can bound the number of real roots of projection polynomials in i variables by m_id_i. If we lifted with respect to all these projection polynomials, the cell count would be bounded by

$$\prod_{i=1}^{n}[2m_id_i+1]=\prod_{s=0}^{\ell}\left[2\left(2^sm2^{2^s-1}d^{2^s}\right)+1\right]\cdot\prod_{r=1}^{n-\ell-1}\left[2\left(2^{2^r\ell}m^{2^r}2^{2^{\ell+r}-1}d^{2^{\ell+r}}\right)+1\right]. \quad (6)$$

Omitting the +1 from each product allows us to calculate the dominant term of the bound explicitly as

$$(2d)^{2^n-1}m^{2^{n-\ell}+\ell-1}2^{\ell 2^{n-\ell}+\ell(\ell-3)/2}. \quad (7)$$

Now we consider the benefit of improved lifting. Start by considering the CAD of $\mathbb{R}^{n-(\ell+1)}$. There can be no reduced lifting until this point and so the cell count bound is given by the second product in (6), which we will denote by †. The lift to $\mathbb{R}^{n-\ell}$ will involve stack generation over all cells, but only with respect to the EC. This can have at most $d_{n-\ell}$ real roots and so the CAD at most $[2d_{n-\ell}+1](\dagger)$ cells.

The next lift, to $\mathbb{R}^{n-\ell-1}$, will lift the sections with respect to the EC, and the sectors only trivially (to produce the same number of cylinders). Hence the cell count bound is $[2d_{n-(\ell-1)}+1]d_{n-\ell}(\dagger)+(d_{n-\ell}+1)(\dagger)$ with dominant term $2d_{n-(\ell-1)}d_{n-\ell}(\dagger)$. Subsequent lifts follow the same pattern and so $2d_nd_{n-1}\ldots d_{n-(\ell-1)}d_{n-\ell}(\dagger)$ is the dominant term in the bound for $\mathbb{R}^n$. This evaluates to give the following result.

Table 1: Projection under operator (1).

Variables	Number	Degree
n	m	d
$n-1$	$2m^2$	$2d^2$
$n-2$	$4m^4$	$8d^4$
$\vdots$	$\vdots$	$\vdots$
$n-r$	$2^{2^{r-1}}m^{2^r}$	$2^{2^r-1}d^{2^r}$
$\vdots$	$\vdots$	$\vdots$
1	$2^{2^{n-2}}m^{2^{n-1}}$	$2^{2^{n-1}-1}d^{2^{n-1}}$

Table 2: Projection with (3) ℓ times and then (1).

Variables	Number	Degree
n	m	d
$n-1$	$2m$	$2d^2$
$\vdots$	$\vdots$	$\vdots$
$n-\ell$	$2^\ell m$	$2^{2^\ell-1}d^{2^\ell}$
$n-(\ell+1)$	$2^{2\ell}m^2$	$2^{2^{\ell+1}-1}d^{2^{\ell+1}}$
$\vdots$	$\vdots$	$\vdots$
$n-(\ell+r)$	$2^{2^r\ell}m^{2^r}$	$2^{2^{\ell+r}-1}d^{2^{\ell+r}}$
$\vdots$	$\vdots$	$\vdots$
1	$2^{2^{(n-1-\ell)}\ell}m^{2^{n-1-\ell}}$	$2^{2^{n-1}-1}d^{2^{n-1}}$

THEOREM 7. *Consider the CAD of $\mathbb{R}^n$ produced using Algorithm 1 in the presence of ECs in the top ℓ variables. The dominant term in the bound on the number of cells is*

$$2\prod_{s=0}^{\ell}\left[2^{2^s-1}d^{2^s}\right]\prod_{r=1}^{n-\ell-1}\left[2\left(2^{2^r\ell}m^{2^r}2^{2^{\ell+r}-1}d^{2^{\ell+r}}\right)\right]=(2d)^{2^n-1}m^{2^{n-\ell}-2}2^{\ell 2^{n-\ell}-3\ell}. \quad (8)$$

The bound in Theorem 7 is strictly less than the one in Theorem 5. The double exponent of m has decreased by the number of ECs; the result of the improved projection in (7). Improved lifting reduced the single exponents further still.

6. CONCLUSIONS AND FUTURE WORK

We have explained how the existing theory for CAD projection using ECs can also be leveraged for significant savings in the lifting phase. We can reduce both the projection polynomials used for lifting and the cells over which stacks are generated. We have formalised these ideas in Algorithm 1, verified their use in Theorem 4, and demonstrated the benefit with a worked example and complexity analysis.

A key question is how to best deal with non-primitive ECs? Consider $\phi := zy=0\wedge\varphi$. Under ordering $\cdots\succ z\succ y\succ\ldots$ the EC $zy=0$ is not primitive, so Algorithm 1 cannot use it. We may be tempted to take $E=\{z\}$ as the primitive part, project with operator (2) and include the content y in the first projection. The CAD of $(y,\ldots)$-space would be sign-invariant for y and thus the CAD of $(z,y,\ldots)$-space truth invariant for the EC (over admissible cells). But we can no longer say only sections are admissible for the next lift as there may be cells with $z\neq 0$ and $y=0$. We could instead lift over all cells. Alternatively we might rewrite ϕ as $\phi:=(z=0\wedge\varphi)\vee(y=0\wedge\varphi)$, so each clause has its own EC. The theory of truth-table invariant CADs [4, 5] is designed to deal with such input, but would require its own extension to use beyond the first projection. Of course, this extension would also be valuable in its own right.

Acknowledgements

Thanks to the the referees for their helpful comments. This work was supported by EPSRC grant: EP/J003247/1.

7. REFERENCES

[1] D. Arnon, G.E. Collins, and S. McCallum. Cylindrical algebraic decomposition I: The basic algorithm. *SIAM J. of Computing*, 13:865–877, 1984.

[2] S. Basu, R. Pollack, and M.F. Roy. Algorithms in Real Algebraic Geometry. (Volume 10 of Algorithms and Computations in Mathematics). Springer-Verlag, 2006.

[3] R. Bradford, C. Chen, J.H. Davenport, M. England, M. Moreno Maza, and D. Wilson. Truth table invariant cylindrical algebraic decomposition by regular chains. In *Computer Algebra in Scientific Computing* (LNCS 8660), pages 44–58. Springer, 2014.

[4] R. Bradford, J.H. Davenport, M. England, S. McCallum, and D. Wilson. Cylindrical algebraic decompositions for boolean combinations. In *Proc. ISSAC '13*, pages 125–132. ACM, 2013.

[5] R. Bradford, J.H. Davenport, M. England, S. McCallum, and D. Wilson. Truth table invariant cylindrical algebraic decomposition. *Submitted for Publication*. Preprint: `arXiv:1401.0645`, 2015.

[6] R. Bradford, J.H. Davenport, M. England, and D. Wilson. Optimising problem formulations for cylindrical algebraic decomposition. In *Intelligent Computer Mathematics* (LNCS 7961), pages 19–34. Springer Berlin Heidelberg, 2013.

[7] C.W. Brown. QEPCAD B: A program for computing with semi-algebraic sets using CADs. *ACM SIGSAM Bulletin*, 37(4):97–108, 2003.

[8] C.W. Brown. Constructing a single open cell in a cylindrical algebraic decomposition. In *Proc. ISSAC '13*, pages 133–140. ACM, 2013.

[9] C.W. Brown, M. El Kahoui, D. Novotni, and A. Weber. Algorithmic methods for investigating equilibria in epidemic modelling. *J. Symbolic Computation*, 41:1157–1173, 2006.

[10] C.W. Brown and S. McCallum. On using bi-equational constraints in CAD construction. In *Proc. ISSAC '05*, pages 76–83. ACM, 2005.

[11] C. Chen, M. Moreno Maza, B. Xia, and L. Yang. Computing cylindrical algebraic decomposition via triangular decomposition. In *Proc. ISSAC '09*, pages 95–102. ACM, 2009.

[12] G.E. Collins. Quantifier elimination by cylindrical algebraic decomposition – 20 years of progress. In *Quantifier Elimination and Cylindrical Algebraic Decomposition*, pages 8–23. Springer-Verlag, 1998.

[13] J.H. Davenport, R. Bradford, M. England, and D. Wilson. Program verification in the presence of complex numbers, functions with branch cuts etc. In *Proc. SYNASC '12*, pages 83–88. IEEE, 2012.

[14] J.H. Davenport and J. Heintz. Real quantifier elimination is doubly exponential. *J. Symbolic Computation*, 5(1-2):29–35, 1988.

[15] A. Dolzmann, A. Seidl, and T. Sturm. Efficient projection orders for CAD. In *Proc. ISSAC '04*, pages 111–118. ACM, 2004.

[16] M. England, R. Bradford, C. Chen, J.H. Davenport, M. Moreno Maza, and D. Wilson. Problem formulation for truth-table invariant cylindrical algebraic decomposition by incremental triangular decomposition. In *Intelligent Computer Mathematics* (LNAI 8543), pages 45–60. Springer, 2014.

[17] M. England, D. Wilson, R. Bradford, and J.H. Davenport. Using the Regular Chains Library to build cylindrical algebraic decompositions by projecting and lifting. *Mathematical Software – ICMS 2014* (LNCS 8592), pages 458–465. Springer Heidelberg, 2014.

[18] M. Erascu and H. Hong. Synthesis of optimal numerical algorithms using real quantifier elimination (Case Study: Square root computation). In *Proc. ISSAC '14*, pages 162–169. ACM, 2014.

[19] I.A. Fotiou, P.A. Parrilo, and M. Morari. Nonlinear parametric optimization using cylindrical algebraic decomposition. In *Proc. CDC-ECC '05*, pages 3735–3740, 2005.

[20] H. Hong. An improvement of the projection operator in cylindrical algebraic decomposition. In *Proc. ISSAC '90*, pages 261–264. ACM, 1990.

[21] Z. Huang, M. England, D. Wilson, J.H. Davenport, L. Paulson, and J. Bridge. Applying machine learning to the problem of choosing a heuristic to select the variable ordering for cylindrical algebraic decomposition. In *Intelligent Computer Mathematics* (LNAI 8543), pages 92–107. Springer, 2014.

[22] H. Iwane, H. Yanami, H. Anai, and K. Yokoyama. An effective implementation of a symbolic-numeric cylindrical algebraic decomposition for quantifier elimination. In *Proc. SNC '09*, pages 55–64, 2009.

[23] S. McCallum. An improved projection operation for cylindrical algebraic decomposition. In *Quantifier Elimination and Cylindrical Algebraic Decomposition*, pages 242–268. Springer-Verlag, 1998.

[24] S. McCallum. On projection in CAD-based quantifier elimination with equational constraint. In *Proc. ISSAC '99*, pages 145–149. ACM, 1999.

[25] S. McCallum. On propagation of equational constraints in CAD-based quantifier elimination. In *Proc. ISSAC '01*, pages 223–231. ACM, 2001.

[26] S. McCallum and C.W. Brown. On delineability of varieties in CAD-based quantifier elimination with two equational constraints. In *Proc. ISSAC '09*, pages 71–78. ACM, 2009.

[27] L.C. Paulson. Metitarski: Past and future. In *Interactive Theorem Proving* (LNCS 7406), 1–10. Springer, 2012.

[28] A. Seidl. Cylindrical decomposition under application-oriented paradigms. PhD Thesis (University of Passau, Germany), 2006.

[29] A. Strzeboński. Cylindrical algebraic decomposition using validated numerics. *J. Symbolic Computation*, 41(9):1021–1038, 2006.

[30] A. Strzeboński. Cylindrical algebraic decomposition using local projections. In *Proc. ISSAC '14*, pages 389–396. ACM, 2014.

[31] D. Wilson, M. England, J.H. Davenport, and R. Bradford. Using the distribution of cells by dimension in a cylindrical algebraic decomposition. *Proc. SYNASC '14*, pages 53–60. IEEE, 2014.